INTERLEAVED PROCESSING SYSTEM FOR PROCESSING FRAMES WITHIN A NETWORK ROUTER

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates generally to systems for processing routing and filtering information FOR each packet received at a router node of a data transmission network, and in particular to an interleaved processing system based upon a tree lookup structure in a network router.

2. Description of the Related Art:

Today, data communication systems are based upon data transmission networks wherein routers are used to link remote sites. Routing is a major bottleneck in such systems primarily due to the overhead processing time and the additional memory capacity required for routing. One of the primary routing functions entails determining a particular routing path for a packet or frame across the network using specific protocols. This path determination is based on a variety of metrics such as the delay introduced by the network or the link cost. In addition, this determination takes into account other rules generically called filtering, such as communication restrictions or priority criteria.

Another important routing function is packet forwarding which entails processing of inbound data packets and the subsequent forwarding of these data packets to the

appropriate outbound destination in accordance with a
destination address within the packet header.

Both the determination of the routing path and packet
forwarding based upon the destination address field in the
packet header, are performed within the same routing
device. Nevertheless, new techniques tend to exploit the
difference between these functions, thus separating the
corresponding operations. For example, a single routing
path processing unit could support several packet
forwarding units.

Since the routing processing time is relatively high
and varies from one routing computation to another, it is
difficult to support multiple time sensitive applications
such as multimedia. For both the filtering and packet
forwarding processing, memory searches consume considerable
time. Within the routing processing context, "searching"
entails retrieving routing information encoded within a
predetermined bit pattern of an packet address header. In
particular, the destination of a data packet typically
corresponds to such a bit pattern within the destination
address field of the packet header. The required search
involves comparing a portion of the destination address bit
pattern to a predetermined bits sequence, or "keys", that
identify appropriate routing information. Efforts have been
made to optimize the speed of such comparison based
searches (often referred to as prefix matching searches) by
using parallel processing, but this method admits its own
limitations.

In addition to packet forwarding, a typical packet
routing cycle includes a filtering process that is
performed with respect to a source address field in the

packet header. Today, the routing process and the filtering process are either performed sequentially utilizing a single processing entity, or are performed simultaneously utilizing two separate processing units. For each process, there are two phases which are repeated until the end of the process. These processes are typically referred to as an instruction loading phase and an instruction processing phase. It should be noted that, in classical systems, the loading and processing phases are very close in duration.

The routing function for determining an outgoing destination node to which a packet will be forwarded, typically utilizes a longest prefix matching (LPM) algorithm that is generally implemented in a tree structure. The filtering process may also utilize this same type of process and algorithm but in a separate tree structure. The trees utilized for routing and filtering are implemented in a memory structure containing an instruction at each tree node, wherein each instruction provides a link to sub-nodes. The tree structure is thus traversed starting from a tree root down to a leaf node that contains an instruction which provides either the routing information or the filtering rules. This last instruction is very often provided in an indirect manner such as by an address value that corresponds to the field that contains the routing, or by filtering information for each leaf node. For further information regarding the nature of LPM searching, reference is made to the article "Routing on longest-matching prefixes", IEEE/ACM transactions on networking, vol. 4, n_1, February 1996, pages 86-97 which is incorporated herein by reference.

Systems currently available for processing the routing
function or the filtering function using a tree structure
require considerable memory overhead for storing the
information in each node, and also consume considerable

5   processing overhead for the requisite large number of
memory accesses. From the foregoing, it can be appreciated
that a need exists for a processing system that enables an
optimized the processing system and method for performing
both routing and filtering functions within a router.

10

SUMMARY OF THE INVENTION

A system and method for performing interleaved packet processing in a network router are disclosed herein. A packet to be routed includes a source address bit pattern and a destination address bit pattern that are each processed by a task processor in accordance with a data tree. The data tree includes multiple nodes linked by branches wherein an instruction that is associated with each node within the data tree is utilized for determining which branch is to be taken in accordance with the source address bit pattern or the destination address bit pattern. A first bank of registers is utilized to load an instruction to be executed by said task processor at each node of the data tree in accordance with the source address bit pattern. A second bank of registers is utilized for loading an instruction to be executed by the task processor at each node of the data tree in accordance with the destination address bit pattern. A task scheduler enables the first bank of registers to transfer an instruction loaded therein for processing by the task processor only during even time cycles and for enabling the second bank of registers to transfer an instruction loaded therein for processing by the task processor only during odd time cycles.

All objects, features, and advantages of the present invention will become apparent in the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Fig. 1 illustrates a tree structure utilized with the interleaved packet processing performed in accordance with a preferred embodiment of the present invention;

Fig. 2 is a block diagram depicting an interleaved packet processing system in accordance with a preferred embodiment of the present invention;

Fig. 3 is a timing diagram illustrating interleaved tasks performed by a task processor in accordance with a preferred embodiment of the present invention;

Fig. 4 is a detailed block diagram depicting an interleaved packet processing system in accordance with a preferred embodiment of the present invention; and

Fig. 5 illustrates a representative format of an lookup instruction utilized in the interleaved packet processing system of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures wherein like reference numerals refer to like and corresponding parts throughout, and in particular with reference to **Fig. 1**, there is illustrated a tree structure utilized with the interleaved packet processing performed in accordance with a preferred embodiment of the present invention. As depicted in **Fig. 1**, a top level **A** of a tree **100**, is commonly referred to as the "root" node. Intermediate nodes, referred to alternately as "branches", are represented by letters **B** to **K**. End nodes ("leafs") represent a corresponding instruction for a packet such as the route for the packet to follow such as **RT0** to **RT5**. For the case in which tree **100** is a routing tree for performing destination address searching, the result of a leaf instruction may be to accept a routing instruction for the packet. If tree **100** is a filter check tree, the result of the tree instruction search may be to determine relevant network routing parameters such as the priority of the packet. Additional filtering decision may be performed utilizing a higher level protocol type.

The left part of tree **100** starting from **A** and proceeding to node **B**, etc. represents a classical binary tree search in which one bit position per unit time is analyzed. As illustrated on the left part of tree **100**, the left branch is selected and traversed when the result of the instruction contained in each branch node is a binary 0 and the right branch is selected when the instruction yields a binary 1.

The right side of tree **100** starting from **A** and proceeding to node **C**, etc. represents a tree search including some binary elements and other more complex elements. For example, node **K** includes four branches that are selectable based on a two-bit analysis and node **G** which has branches that are selectable based on a comparison done on several long binary patterns (5 bits in the example).

For the embodiments depicted by the figures herein, it is assumed that multiple 16-bit instruction words are associated with each node for routing or filtering processing. The selection of a 16-bit instruction is based on the current state of the art in which 16-bit instruction words are common. It should be noted, however, that for complex decision methods or for cases in which more than two branches are involved, instructions can have different lengths without departing from the spirit or scope of the present invention.

With reference now to **Fig. 2**, there is illustrated a block diagram depicting an interleaved packet processing system in accordance with a preferred embodiment of the present invention. Specifically, an interleaved packet processing system **200** includes a bi-task processing engine **10** connected to an external memory **12** by a standard address bus, data bus and control bus represented schematically by a bus **14**. External memory **12** stores instructions that are executed at each node of the tree and is divided into two sub-memories **12-1** and **12-2**. Sub-memory **12-1** contains normal size (16-bit) instructions that are executed only in response to particular predefined values on most significant bit (MSB) of a binary address sequence, while area **12-2** contains dual (double) size instructions. In

accordance with a preferred embodiment of the present
invention, external memory 12 contains the instructions of
two independent trees.

As further depicted in **Fig. 2**, processing engine 10
includes a main task processor 16 that can be either a
finite state machine or a nano-processor and a task
scheduler 18 that generates "thread" clocking composed of
successive alternate even and odd time cycles and enables
processing activity and loading for each task. Task
scheduler 18 is connected to task processor 16 to initiate
a node process. Task scheduler 18 is also connected to
registers banks, bank **A** 20 and bank **B** 22 via activation
lines 24 and 26.

Banks 20 and 22 contain respectively the instructions
associated with each node in a given tree. Activation
signal carried from task scheduler 18 on activation lines
24 and 26 are utilized for loading an instruction from
external memory 12 to one of banks 20 or 22 via external
bus 14. Activation signals on activation lines 24 and 26
also activate the transfer of an instruction from the other
bank (the bank not being loaded from external memory 12) to
task processor 16 via an internal bus 28 for processing the
instruction. At any given time, a bank has only one of the
above access valid signals carried on bus 28 while the
other bank has access to bus 14. This bus connection is
inverted at each edge of the thread clock allowing the
process of an instruction on one tree while an instruction
of the other tree is loaded into its corresponding bank.
In a preferred embodiment of the present invention, one of
banks 20 or 22 is associated with a source address tree

while the other bank is associated with a destination address tree in an address lookup mechanism.

Multiple temporary registers **30** within processing engine **10** contain information for each task that is maintained between two consecutive processing periods of processor **16**. Temporary registers **30** are particularly useful when the processing of an instruction is split into two times cycles of the clock within task scheduler **18**. The address bit patterns that are processed by instructions loaded into processor **16** are provided by a system bus **32** into a pattern register **A 34** and a pattern register **B 36**.

Turning now to **Fig. 3**, there is illustrated a timing diagram depicting the interleaving of tasks performed by task processor **16** as per the direction of task scheduler **18** in accordance with a preferred embodiment of the present invention. As shown in **Fig. 3**, the thread clock maintained by task scheduler **18** has a rising edge when process **A** is activated and a falling edge when the process **B** starts. The instruction structure encompassed by processing engine **10** is designed to operating in temporal increments corresponding exactly to the duration of each clock cycle which is constant so that there is no need to check whether process **A** is completed or not, and similarly for process **B**.

While task **A** is processed, a next instruction for task **B** is loaded into register bank **B**. Similarly, while task **B** is being processed, a next instruction for task **A** is loaded into bank **A**. The result is a continual interleaving process in which task **A** and task **B** until each process reaches a leaf node.

If there is a need to process a longer instruction
(stored in the second memory area **12-2**) that cannot fit
within a single cycle, temp register **30** within processing
engine **10** enables utilization of two cycles for processing
such an instruction. **Fig. 3** depicts a dual loading of a
long instruction during task **A**. As shown in **Fig. 3**,
processing for task **A** is delayed until the full loading is
completed over two task **A** loading cycles that are
interleaved with process cycles as per the scheduling
provided by task scheduler **18**. It is also possible to
start part of the process and to set **A** or **B** state bit
(illustrated in **Fig. 4**) when the cycle ends and before the
end of the cycle, to store intermediate state or results
into the temp register **30** dedicated to the corresponding
task.

The dual processing of an instruction for task **A** is
also depicted in **Fig. 3** for the case in which a single size
instruction halts the loading during one thread until the
instruction is fully processed. This mechanism requires
the use of temp register **30**, as task processor **18** will
process task **B** in between processing intervals for the
processing of task **A**.

With reference now to **Fig. 4**, there is illustrated a
detailed block diagram depicting interleaved packet
processing system **200** in accordance with a preferred
embodiment of the present invention. At the beginning of a
packet address processing interval, bank **A 20** is loaded by
a bit pattern that provides as a next instruction, the root
address for tree **A**. The stored address for the next
instruction (the root address in the first step but

contained in the instruction for further steps) is provided by task processor **16** via NEXT ADD bus **40** and driver **42** to bank **A**.

The address contents are loaded into an ADD (address) register **44** via bank data bus **46** as "InsAD" (Instruction Address) in response to a BKRD Bank read command activated by a NEXTCMD signal from task processor **16** on line **48**. The NXTCMD signal instructs ADD register **44** to load this next address for instruction. NEXTCMD includes the index (IX) (described in **Figure** 5) used to increment the next address for instruction. ADD register **44** is a register/counter loaded by "InsAD" which is the address of the first word of the instruction to be loaded and whose value is incremented by an integrated counter (not depicted) to load the other words of the instruction.

When a rising edge is detected on CKA line **50** from task scheduler **18**, the first address corresponding to the initial state of the counter is presented on external add bus **14** with a ext read command on line **51** in order to access the external memory. A driver **52** is activated by Ext read command on line **51** and a driver **42** is also opened to present the least significant bits (LSBs) of the address to load the memory field into the corresponding register of bank **A 20** with a bank write command BKWR. As the first address indicates whether the instruction is a single instruction (located in area **12-1**) or a dual instruction (located in area **12-2**), it is possible to stop the counter at the end of the cycle on the last word that should be downloaded.

In case of a dual size instruction, a bit DUAL is set
within ADD register **44** which prevents task scheduler **18**
through CKA line **50** from reloading the address from bank **A**
to ADD register **44** thus allowing the counter to continue to
load the remaining words of the instruction into bank **A**
during the next cycle. The size of bank **A** is designed to
accept a dual size instruction. For each word loaded from
memory in response to a read signal and a chip select
signal, ADD register **44** selects the appropriate register in
bank **A** by using the LSB bits of the counter value on bank
add bus **54** as the register address to load the instruction
word in the appropriate bank register in response to a bank
write (BKWR) to this location. Upon occurrence of the next
CKA rising edge, the DUAL bit is tested. If the DUAL bit is
set, the counter continues and loads the additional
instruction words until the counter reaches its limit. If
the DUAL bit is not set, ADD register **44** loads a BKWR Read
command into the register of bank **A** selected by NEXT ADD on
bus **40** from task processor **16**. The loaded BKWR Read
command allows the InsAd Instruction Address to be
delivered on bank data bus **46** and then stored in ADD
register **44**.

When the instruction is loaded in bank **A 20**, task
scheduler **18** delivers a CKA signal to inform task processor
16 to fetch the first instruction word in bank **A**. The
first register address is delivered by task processor **16**
via add bus **56**. A read is then performed on bank **A** in
response to a SEL A command on line **58**, and the first
instruction word `provided from register bank **A** on data bus
60 is utilized by task processor **16** which can then execute
the instruction based on the pattern associated with the

loaded instruction to compare it with pattern **A** in register
34.   The result of such a comparison may be to load other
instruction words from bank A using the same mechanism.   At
the end of a processing cycle corresponding to completion
of a single clock cycle, either the process is not
completed and temporary results are stored in Temp Register
30-1 for task **A** or the process of the current instruction
is finished.   Temp register **30-2** is reserved for temporary
storage of task **B**.

In the latter case, when the instruction process is
completed, the position (address in bank **A**) corresponding
to the address of the next instruction in bank **A** is put on
next ADD bus **40** using a few bits (3 for eight register
positions in bank **A**) in order to provide the address for
the next loading to ADD register **44**.

In the former case, in which the instruction process
is not completed, two 1-bit **A** state and **B** state registers
within task processor **16** are each utilized to define the
state at the end of the processing cycle that is used when
more than one cycle is processed.   The setting of the
single bit in **A** state and **B** state registers indicates
whether the next instruction to be processed is a
continuation of the previous processing activity or a new
instruction.

Turning now to **Fig. 5**, there is illustrated a
representative format of an lookup instruction utilized in
the interleaved packet processing system of the present
invention.   As shown in **Fig. 5**, each instruction contains
three main fields.   The first of these fields includes the
instruction itself that is executed by the task processor.

The second field in a comparison field which contains the pattern previously stored within the external memory (in a table, for example) to compare with the **A** or **B** pattern that correspond to a source and destination address bit pattern at the current point of the address processing analysis. Finally, the lookup instruction includes a next address field that contains the addresses for the possible next instructions.

The instruction itself is generally defined in a single word. The contents of the instruction field define the mode of analysis to perform such as one bit, two bits or three bits full comparison resulting in two, four, or eight branches, or, whether a multi-bit pattern comparison should be performed using further comparison fields. The instruction field contents also include fields defining the number of elements (bits) within the comparison field, and the number of bits in the next address field that defines the size of the instruction and informs the processor of whether or not an instruction is fully loaded.

In full comparison mode, additional fields are defined that identify the next address to use for each output case that can be a direct value or an indexed value. There is one such sub-field for each possible branch. **Fig. 5** illustrates a case in which **4** branches corresponding to a two bits full comparison: branches **00**, **01**, **10** and **11**.

The index (IX) is a 2-bit field that indicates the actual address value based on the address given as a base address. A value of **00** for IX indicates that the address is the address of the indicated next add field, while **01** instructs to increment by one the indicated next add field.

An IX value of **10** instructs to increment by 2, and **11** to increment by 3. A single next add field allows for pointing onto up to 4 different instruction elements in memory thus reducing the size of the instruction itself.

The comparison field stores, if any, the pattern(s) to compare to the bits starting at the current position in the **A** or **B** pattern field. For each pattern, a sub-field indicates the length of the pattern (Nbbits), a possible mask (Pattern Mask), the next address sub field (direct or indexed) to use or next comparison to perform when match or not match. The index method is the same as what is defined in the instruction field. It should be noted that, when the link is performed on another comparison field, the index field (IX) is irrelevant.

The next address field contains the list of addresses of the nodes connected to the branches of the current node. Consecutive addresses may be used but cannot always be used as in case of multiple branches, wherein an addresses may be followed by a single instruction and while others are followed by a dual instruction.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.